Evaluating Network Embeddings through the Lens of Community Structure

Jason Barbour[1,2], Stephany Rajeh[3,4], Sara Najem[2,5], and Hocine Cherifi[6]

[1] Graduate Program in Computational Science, American University of Beirut, Beirut, Lebanon
[2] Department of Physics, American University of Beirut, Beirut, Lebanon
[3] LIP6 CNRS, Sorbonne University, Paris, France
[4] Efrei Research Lab, EFREI Paris, Villejuif, France
[5] Center for Advanced Mathematical Sciences, American University of Beirut, Beirut, Lebanon
[6] ICB UMR 6303 CNRS, University of Burgundy, Dijon, France

Abstract. Network embedding, a technique that transforms the nodes and edges of a network into low-dimensional vector representations while preserving relevant structural and semantic information, has gained prominence in recent years. Community structure is one of the most prevalent features of networks, and ensuring its preservation is crucial to represent the network in a lower-dimensional space accurately. While the core objective of network embedding is to bring related nodes in the original network close together in a lower-dimensional space, common classification metrics overlook community structure preservation. This work addresses the need for a comprehensive analysis of network embedding algorithms at the community level. On a set of synthetic networks that span strong to weak community structure strengths, we showcase the variability in the performance of network embedding techniques across mesoscopic metrics. Additionally, we highlight that the mesoscopic metrics are not highly correlated with the classification metrics. The community structure can further diminish the correlation as its strength weakens.

Keywords: Network Embedding, Community Structure, Evaluation Metrics

1 Introduction

Networks often exhibit a modular structure, where nodes cluster into communities with shared characteristics or functions [1]. Understanding these community structures is crucial for various applications, from recommendation systems to the optimal spread of information and disease control [2–10]. With network sizes increasingly increasing, generating lower order representation, known as network embedding, has gained significant attention in recent years [11]. This technique transforms networks into low-dimensional vector representations.

While certain techniques are designed to explicitly maintain or enhance the community structure through the embedding process, others may not consider community structure preservation a primary objective. Nonetheless, one of the fundamental goals of all network embedding techniques is to project the similarity of the nodes of the original network onto the lower-dimensional space.

Further, network embedding techniques are commonly evaluated through classification metrics [11]. Nonetheless, these metrics are agnostic about the community structure: they do not indicate whether it is well preserved after the embedding process. In other words, they offer information about the overall quality of results but do not reveal the fine-grained details of community structure within a network.

Consequently, there is a need for a comprehensive comparative analysis of network embedding algorithms from a modular perspective. This paper analyzes the performance of the most prominent network embedding algorithms on controlled synthetic networks.

The rest of the paper is organized as follows. Section 2 overviews the fundamental concepts of network embedding and introduces the mesoscopic evaluation metrics. Section 3 presents the synthetic modular network generation. It details the experimental setup and evaluation metrics for comparing these algorithms and the voting model used in ranking these algorithms. Section 4 presents the results of our comparative analysis and discusses the findings and their implications. Finally, Section 5 concludes the paper.

2 Background

The landscape of network embedding algorithms is notably diverse. To ensure that we encompass a spectrum of approaches, we include ten widely recognized methods that span random walks, matrix factorization, and deep learning [12]. This diversity should allow us to understand the challenges and opportunities of different network embedding strategies. We briefly describe these algorithms, highlighting their specificity. It is worth highlighting that the DeepWalk, Node2Vec, Walklets, M-NMF, and M-GAE algorithms explicitly indicate maintaining the community structure as they acquire node representations.

2.1 Random-walk-based Methods

- **DeepWalk:** applies techniques from deep learning to learn node embeddings by treating network walks as sentences and using Skip-gram models to capture the context of nodes. It uses random walks to explore the network and capture local neighborhood information [13].
- **Node2Vec:** extends DeepWalk by introducing a biased random walk strategy that explores both breadth-first and depth-first neighborhood structures. It allows for fine-tuning exploration behavior, making it more versatile for different network structures [14].

- **Diff2Vec:** incorporates differential information from network snapshots over time to capture evolving node embeddings. It is particularly suitable for dynamic networks where nodes and their connections change over time [15].
- **Walklets:** is a variation of DeepWalk that leverages multiple lengths of random walks to capture local and global network structures. It creates embeddings by considering various context sizes, effectively capturing hierarchical relationships [16].

2.2 Matrix Factorization-based Methods

- **Modularity-Normalized Matrix Factorization (M-NMF):** M-NMF optimizes a modularity-based objective function to learn embeddings that preserve community structure. It directly incorporates community information into the embedding process, emphasizing preserving community properties [17].
- **Laplacian Eigenmaps (LEM):** is a spectral embedding method that utilizes the eigenvalues and eigenvectors of the Laplacian matrix to map nodes to a low-dimensional space. It emphasizes preserving the pairwise distances between nodes, which can highlight the underlying geometric structure of the network [18].
- **Randomized Network Embedding (RandNE):** employs randomized matrix factorization to generate node embeddings while preserving global and local network properties. It introduces randomness in the factorization process, which can lead to more diverse embeddings [19].
- **Boosted Network Embedding (BoostNE):** leverages ensemble learning techniques to combine multiple embeddings generated by different methods, enhancing the overall performance. It focuses on improving embedding quality through ensemble techniques and is versatile in incorporating various base embedding methods [20].
- **Network Matrix Factorization (NetMF):** factorizes the network's adjacency matrix to obtain embeddings that capture higher-order proximity patterns. It emphasizes capturing different types of proximities in the network, such as Katz similarity, which goes beyond traditional pairwise node relationships [21].

2.3 Deep learning-based Method

- **Modularity-Aware Graph Autoencoder (M-GAE):** utilizes autoencoder architectures to learn embeddings by reconstructing the adjacency matrix or other network-related properties. It introduces a novel regularizer inspired by modularity and can adapt to various reconstruction objectives, allowing it to capture different aspects of network structure [22].

2.4 Evaluation Metrics

The classification metrics used in the literature to evaluate the quality of the embedding algorithms are mainly grounded in information theory and heavily

used by the machine learning community, which include the adjusted mutual information score (AMI), normalized mutual information score (NMI), adjusted random score (ARI), Micro-F1 score, and Macro-F1 score, which are not necessarily community-aware evaluators [23]. Here we propose a complementary set of metrics to assess the quality of these algorithms. These below-described mesoscopic metrics are used to evaluate the quality of the embedding techniques in preserving the community structure. The latter, denoted by C, is defined as follows: for an undirected unweighted graph $G(V, E)$, where V is the set of nodes and $E \subseteq V \times V$ is the set of edges, $C = \{c_1, c_2, ..., c_q, ..., c_{|C|}\}$ where c_q is q-th community, m_{c_q} and n_{c_q} are the total number of links and nodes inside community c_q, respectively, and $\mid C \mid$ is the total number of communities. A node i in G has a total degree of $k_i^{tot} = k_i^{intra} + k_i^{inter}$ where k_i^{intra} denotes its intra-community links and k_i^{inter} denotes its inter-community links. The mesoscopic metrics are calculated for each community in the network and then averaged over all the communities. We denote each evaluation metric for each community c_q as $f(c_q)$. In the context of this study, we employ a set of nine mesoscopic metrics, where seven are defined within our work. The internal degree and community size distributions are inherently self-explanatory through their nomenclature:

- **Internal distance:** is the average shortest distance of nodes inside a given community $f(c_q)$:

$$f(c_q) = \sum_{i,j \in c} \frac{d(i,j)}{n_{c_q}(n_{c_q} - 1)}$$

  where $d(i, j)$ is the shortest path from node i to node j.
- **Internal density:** is the edge density inside a given community c_q:

$$f(c_q) = \frac{2m_{c_q}}{n_{c_q}(n_{c_q} - 1)}$$

- **Maximum-out degree fraction (Max-ODF):** is based on the inter-community links of a node that possesses the highest inter-community links in its community c_q:

$$f(c_q) = max_{(i \in c_q)} \frac{k_i^{inter}}{k_i^{tot}}$$

- **Average-out degree fraction (Average-ODF):** is based on the inter-community links of all the nodes in the community c_q they belong to:

$$f(c_q) = \frac{1}{n_{c_q}} \sum_{i \in c_q} \frac{k_i^{inter}}{k_i^{tot}}$$

- **Flake-Out degree fraction (Flake-ODF):** is based on the percentage of nodes in community c_q that have more inter-community links than intra-community links:

$$f(c_q) = \sum_{i \in c_q} \frac{|f_i|}{n_{c_q}}$$

where f_i is the fraction of nodes having $k_i^{inter} \geq k_i^{intra}$.

– **Embeddedness:** quantifies the intra-community links of a node. It is the opposite of Average-ODF. It reaches a value of 1 if the nodes in a given community c_q only have intra-community links (i.e., all neighbors are in the same community):

$$f(c_q) = \frac{1}{n_{c_q}} \sum_{i \in c_q} \frac{k_i^{intra}}{k_i^{tot}}$$

– **Hub dominance:** is based on the intra-community links of a node that has the highest intra-community links in its community c_q:

$$f(c_q) = max_{(i \in c_q)} \frac{k_i^{intra}}{n_{c_q}(n_{c_q} - 1)}$$

3 Experimental Setup and Evaluation

This section describes the fundamental steps of the experimental setup employed to evaluate the network embedding algorithms' efficacy in maintaining the community structure.

3.1 Synthetic Network Generation

The ABCD generator offers the ability to change multiple parameters of the network generation process, namely: number of nodes (N), power-law exponent for degree distribution (τ_1), minimum degree (d_{min}), maximum degree (d_{max}), power-law exponent for community size distribution (τ_2), minimum community size (c_{min}), maximum community size (c_{max}) and the mixing parameter (μ) [24]. In what follows, we use it with the below choice for the parameters:

$$\begin{cases} N = 10000 & \tau_1 = 2.7 \\ d_{min} = 3 & d_{max} = 150 \\ \tau_2 = 2.7 & c_{min} = 10 \\ c_{max} = 1000 & \mu = [0.1 \rightarrow 0.7] \end{cases} \tag{1}$$

For our experimental setup, we focus on μ as a control parameter and vary it from 0.1 to 0.7 in increments of 0.1 and thus generate seven corresponding graphs. μ controls the number of edges between clusters where 0 represents the absence of inter-clusters edges and 1 is the other limit of high connectivity where the clusters become highly mixed. ABCD generator provides labeling for each node indicating which community it belongs to, which will be used as the ground truth for the rest of this paper.

3.2 Experimental Setup

Seven main steps are described below to evaluate the network embedding algorithms. A bird's-eye view of the experimental setup is illustrated in Figure 1.

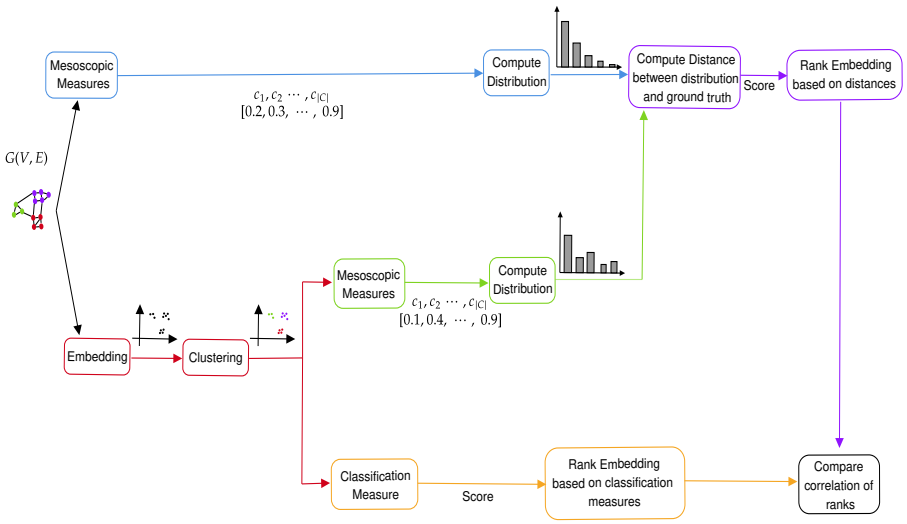


Fig. 1: Flowchart of the experimental setup to evaluate the performance of the network embedding algorithms. Mesoscopic metrics are calculated individually for each of the network's communities.

- **Embedding the graphs:** We use ten different embedding algorithms for each generated network to embed the network into a 128-dimensional space. The algorithms under investigation are: Deepwalk, Node2Vec, Diff2Vec, Walklets, M-NMF, Laplacian Eigenmaps, RandNE, BoostNE, NetMF, and M-GAE[1].
- **Clustering the embeddings:** Once we have the embeddings, we use the K-means clustering algorithm to group them into clusters whose number is defined by the ABCD as it as the number of clusters.
- **Compute the distance between the distributions:** We compute the Kullback–Leibler (KL) divergence [25] between all the above-introduced mesoscopic measures between the ground truth and after the embedding[2], includ-

[1] The embedding algorithms were run on the American University of Beirut (AUB)'s high-performance computers using Intel Xeon E5-2665 CPUs in parallel. The runtime of these algorithms is reported in https://github.com/JasonBarbour-2002/ExploringNetworkEmbeddings in Figure 1 of the supplementary material.

[2] Since most methods rely on a stochastic process, we run each method 30 times and take the average and standard deviation of the score for each measure.

ing the internal degree distribution, community size distribution, internal distance, internal density, Max-ODF, Average-ODF, Flake-ODF, embeddedness, and hub dominance for all communities.
- **Compute the classification metrics:** After clustering, classification metrics including AMI, NMI, ARI, Micro-F1 score, and Macro-F1 score are computed.
- **Ranking the embedding algorithms based on the mesoscopic metrics:** Following the KL-divergence computation between the ground-truth distribution and the distribution of the embedding algorithms after the embedding process, the embedding algorithms are ranked from the most performing (i.e., lowest distance) and least performing (i.e., highest distance)
- **Ranking the embedding algorithms based on the classification metrics:** Following the computation of the classification metrics, the embedding algorithms are ranked from the most performing (i.e., highest magnitude) and least performing (i.e., lowest magnitude).
- **Comparing the correlation of the ranks:** When the ranks of the embedding techniques for all metrics are obtained, the correlation between the mesoscopic and classification metrics is computed.

3.3 Voting Model

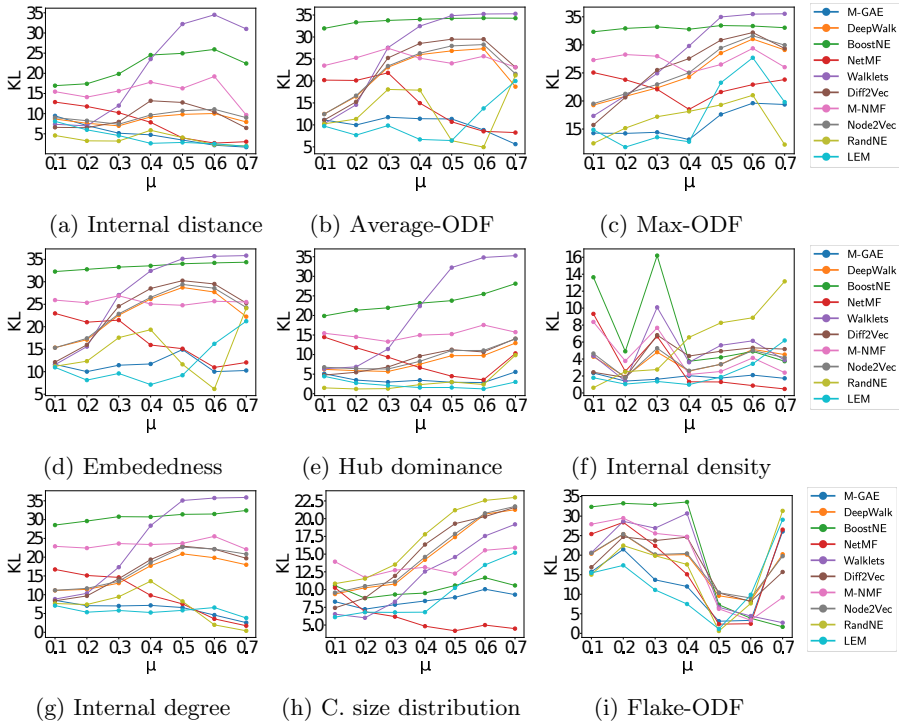
Given the metrics used as performance evaluators, a ranking scheme to report on the overall quality of the algorithms is needed. For this purpose, we used Schulze's voting model [26]. First, we compute the KL-divergence score of the distributions of the mesoscopic metrics versus the ground truth ones, as well as the classification metrics of the network embedding algorithms, which will be the voters, while the candidates are the ranks of each algorithm for each metric. Subsequently, the voting model orders them by comparing candidates in head-to-head matchups, calculating how many voters prefer one candidate. It then finds the candidate that outperforms all others by the largest margin, aiming to identify a broadly preferred winner. In the end, the final ranks of the network embedding algorithms represent a consensus between all the metrics.

4 Results and Discussion

In what follows, we present the performance of each embedding algorithm for the above-listed mesoscopic measures by evaluating the KL divergence between the ground truth distribution of the mesoscopic measure in question and the recovered one.

In Figure 2a, we follow the KL-divergence of the distribution of internal distance. Almost all algorithms consistently have a low KL divergence as μ grows. Beyond $\mu = 0.4$, the performances of Waklets and BoostNE drop significantly as they exhibit an increase in their respective KL-divergences.

As for the Average-ODF, BoostNE stands out with the lowest performance up to $\mu = 0.4$. This is also accompanied by a drop in the performance of Waklets as shown in Figure 2b.



(a) Internal distance (b) Average-ODF (c) Max-ODF

(d) Embededness (e) Hub dominance (f) Internal density

(g) Internal degree (h) C. size distribution (i) Flake-ODF

Fig. 2: KL-divergence between the ground truth distributions of the mesoscopic metrics and those recovered by the different algorithms.

Concerning Max-ODF, embeddedness and internal degree, over the full range of μ, BoostNE is outperformed by the other algorithms as shown in Figures 2c, 2d, and 2g. Interestingly, it is joined by Walklets after the $\mu = 0.4$ value. Hub dominance in Figure 2e follows this trend. However, in this case, the performance of Walklets decreases considerably more than the value of BoostNE.

For internal density in Figure 2f, BoostNE is performing the worst up to $\mu = 0.4$, which becomes similar to the rest of the algorithms. At that point, we see a big decrease in the performance of RandNE.

As for Flake-ODF, all the algorithms seem to be following the same trend where the value of the KL is high for low values of μ, then after the value of $\mu = 0.4$, we see a sharp decline.

In the case of community size distribution, all algorithms' performance decreases around $\mu = 0.4$ as shown in Figure 2h, except for NetMF which increases.

The results of the comparison of the ranks are rendered to a heatmap describing the correlation between the rankings of the classification metrics and the mesoscopic ones for the values of $\mu = 0.1$ and 0.7 are shown in Figures 3 and 4, respectively. We note that the performance of the different algorithms

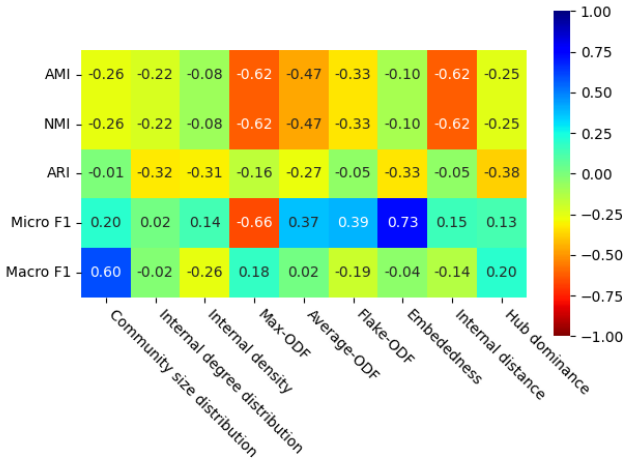


Fig. 3: Correlation between the ranking of the algorithms based on the classification metrics and the mesoscopic metrics for $\mu = 0.1$.

changes with the mixing parameter μ. Particularly, beyond the value of $\mu = 0.4$, some algorithms exhibit a significant drop in performance, which we report in Table 1. In the latter table, we also report the ranking of the algorithms using the classification metrics.

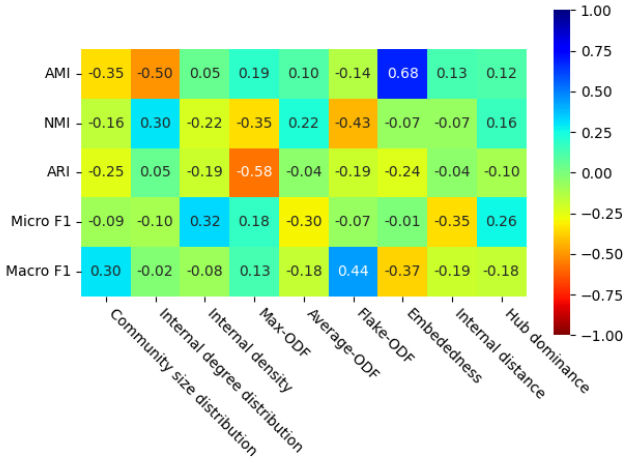


Fig. 4: Correlation between the ranking of the algorithms based on the classification metrics and the mesoscopic metrics for $\mu = 0.7$.

Based on the mesoscopic metrics, denoted as Meso in the table, LEM ranks first while M-GAE ranks second for $\mu \leq 0.4$, which is in total agreement with the performance assessment based on the classification metrics, denoted by CL in the table. On the other hand, for Meso, when $\mu > 0.4$, M-GAE takes the lead, while NetMF ranks second. The latter remains true when the classification metrics. From the above it is obvious that the performance quality is μ-dependent, revealing the effect of mixing. Though Table 1 seems to suggest that there is total agreement between the measures, Figures 3 and 4 shows the mismatch between the mesoscopic and the classification metrics in which we report uncorrelation and decorrelation between the measures.

		Ranks									
μ	Metrics	1^{st}	2^{nd}	3^{rd}	4^{th}	5^{th}	6^{th}	7^{th}	8^{th}	9^{th}	10^{th}
$\mu \leq 0.4$	Meso	LEM	M-GAE	RandNE	NetMF	Deepwalk	Node2Vec	Diff2Vec	Walklets	M-NMF	BoostNE
	CL	LEM	M-GAE	RandNE	Walklets	NetMF	M-NMF Node2Vec		Deepwalk	Diff2Vec	BoostNE
$\mu > 0.4$	Meso	M-GAE	NetMF	RandNE LEM		M-NMF	Deepwalk	Diff2Vec	Node2Vec	BoostNE	Walklets
	CL	M-GAE	NetMF	M-NMF	BoostNE	LEM	Deepwalk	Node2Vec RandNE		Diff2Vec	Walklets
Total	Meso	LEM	M-GAE	RandNE	NetMF	Deepwalk	Node2Vec	Diff2Vec	M-NMF	Walklets	BoostNE
	CL	M-GAE	LEM	NetMF	M-NMF	RandNE	Node2Vec	Deepwalk	BoostNE	Walklets	Diff2Vec

Table 1: The ranking of the embedding algorithms based on Schulze's method for mesoscopic and classification metrics.

It is also worth noting that RandNE, LEM, and NetMF are matrix factorization methods that rely on a distance optimization scheme, while M-GAE is a deep learning-based algorithm. This seems to suggest that the metrics have inherent biases. More precisely, LEM, which relies on evaluating the Laplacian, is expected to rank first for most mesoscopic measures. The reason is the Laplacian's properties, the difference between the degree and the adjacency matrices. In that, it retains information about each node's degree and neighborhood. Moreover, the number of degenerate eigenvectors corresponding to the eigenvalue 0 of the Laplacian represents the number of communities, equivalent to a distance optimization problem [27]. In contrast, the mutual information measures tend to rank M-GAE first. That could be explained by the fact that M-GAE is optimizing on modularity which is related to the mutual information between two nodes. Therefore, it seems like it is a self-looping metric that is optimizing for itself.

In accordance with the outcomes presented here, upon crossing $\mu = 0.4$, M-GAE exhibits superior performance while it is second when $\mu \leq 0.4$ in both classification and mesoscopic metrics. Additionally, LEM ranks first when $\mu \leq 0.4$, ranks fifth with $\mu > 0.4$; NetMF, which ranks fourth and fifth with mesoscopic metrics and classification metrics, respectively, now ranks second when $\mu > 0.4$.

To summarize, LEM demonstrates outstanding performance within a robust community structure, excelling in community-aware and classification metrics. However, as the community structure strength diminishes, its effectiveness prominently declines with classification metrics. The opposite behavior is seen with NetMF. In contrast, M-GAE maintains outperformance across both community-aware and classification metrics regardless of the community structure strength, by ranking either first or second.

5 Conclusion

Preserving network community structure is crucial, and network embedding techniques offer a significant potential. However, the evaluation metrics commonly used in the literature fail to capture this preservation effectively. This study highlights the need for a comprehensive comparison of network embedding algorithms from a modular perspective. Our work is limited to evaluating the effect of the mixing parameter on the embedding quality. Our study specifically aims to determine the adequacy of classification metrics employed in the literature to comprehend the effectiveness of network embeddings. Results reveal that these metrics do not comprehensively reflect the network's community structure, exhibiting a low correlation with community-aware metrics. Furthermore, the efficacy of certain embedding techniques, such as LEM, M-GAE, and NetMF, is influenced by the strength of the community structure. These findings underscore the need for a more attentive approach in evaluating embedding techniques tailored to the specific application.

References

1. Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. *PNAS*, 99(12):7821–7826, 2002.
2. Hocine Cherifi, Gergely Palla, Boleslaw K Szymanski, and Xiaoyan Lu. On community structure in complex networks: challenges and opportunities. *Applied Network Science*, 4(1):1–35, 2019.
3. Keziban Orman, Vincent Labatut, and Hocine Cherifi. An empirical study of the relation between community structure and transitivity. In *Complex Networks*, pages 99–110. Springer Berlin Heidelberg Berlin, Heidelberg, 2013.
4. Naveen Gupta, Anurag Singh, and Hocine Cherifi. Community-based immunization strategies for epidemic control. In *2015 7th international conference on communication systems and networks (COMSNETS)*, pages 1–6. IEEE, 2015.
5. Debayan Chakraborty, Anurag Singh, and Hocine Cherifi. Immunization strategies based on the overlapping nodes in networks with community structure. In *Computational Social Networks: 5th International Conference, CSoNet 2016, Ho*

Chi Minh City, Vietnam, August 2-4, 2016, Proceedings 5, pages 62–73. Springer International Publishing, 2016.
6. Manish Kumar, Anurag Singh, and Hocine Cherifi. An efficient immunization strategy using overlapping nodes and its neighborhoods. In *Companion Proceedings of the The Web Conference 2018*, pages 1269–1275, 2018.
7. Zakariya Ghalmane, Chantal Cherifi, Hocine Cherifi, and Mohammed El Hassouni. Extracting backbones in weighted modular complex networks. *Scientific Reports*, 10(1):15539, 2020.
8. Stephany Rajeh, Marinette Savonnet, Eric Leclercq, and Hocine Cherifi. Interplay between hierarchy and centrality in complex networks. *IEEE Access*, 8:129717–129742, 2020.
9. Stephany Rajeh, Marinette Savonnet, Eric Leclercq, and Hocine Cherifi. Characterizing the interactions between classical and community-aware centrality measures in complex networks. *Scientific reports*, 11(1):10088, 2021.
10. Stephany Rajeh, Marinette Savonnet, Eric Leclercq, and Hocine Cherifi. Comparative evaluation of community-aware centrality measures. *Quality & Quantity*, 57(2):1273–1302, 2023.
11. Mingliang Hou, Jing Ren, Da Zhang, Xiangjie Kong, Dongyu Zhang, and Feng Xia. Network embedding: Taxonomies, frameworks and applications. *Computer Science Review*, 38:100296, 2020.
12. Palash Goyal and Emilio Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151:78–94, 2018.
13. Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proc. of the 20th ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, pages 701–710, 2014.
14. Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proc. of the 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, pages 855–864, 2016.
15. Benedek Rozemberczki and Rik Sarkar. Fast sequence-based embedding with diffusion graphs. In *Complex Networks IX: Proc. of the 9th Conf. on Complex Networks CompleNet 2018 9*, pages 99–107. Springer, 2018.
16. Bryan Perozzi, Vivek Kulkarni, Haochen Chen, and Steven Skiena. Don't walk, skip! online learning of multi-scale network embeddings. In *Proc. of the 2017 IEEE/ACM Int. Conf. on Advances in Social Networks Analysis and Mining 2017*, pages 258–265, 2017.
17. Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. Community preserving network embedding. In *Proc. of the AAAI conf. on artificial intelligence*, volume 31, 2017.
18. Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. *Advances in neural information processing systems*, 14, 2001.
19. Ziwei Zhang, Peng Cui, Haoyang Li, Xiao Wang, and Wenwu Zhu. Billion-scale network embedding with iterative random projection. In *ICDM*, pages 787–796. IEEE, 2018.
20. Jundong Li, Liang Wu, Ruocheng Guo, Chenghao Liu, and Huan Liu. Multi-level network embedding with boosted low-rank matrix approximation. In *Proc. of the 2019 IEEE/ACM int. conf. on advances in social networks analysis and mining*, pages 49–56, 2019.
21. Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and

node2vec. In *Proc. of the eleventh ACM int. conf. on web search and data mining*, pages 459–467, 2018.
22. Guillaume Salha-Galvan, Johannes F Lutzeyer, George Dasoulas, Romain Hennequin, and Michalis Vazirgiannis. Modularity-aware graph autoencoders for joint community detection and link prediction. *Neural Networks*, 153:474–495, 2022.
23. Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: is a correction for chance necessary? In *Proc. of the 26th annual int. conf. on machine learning*, pages 1073–1080, 2009.
24. Bogumił Kamiński, Paweł Prałat, and François Théberge. Artificial benchmark for community detection (abcd)—fast random graph model with community structure. *Network Science*, 9(2):153–178, 2021.
25. Solomon Kullback and Richard A Leibler. On information and sufficiency. *Ann. Math. Stat.*, 22(1):79–86, 1951.
26. Markus Schulze. A new monotonic, clone-independent, reversal symmetric, and condorcet-consistent single-winner election method. *Social choice and Welfare*, 36:267–303, 2011.
27. Gilbert Strang. *Linear algebra and learning from data*. SIAM, 2019.

View publication stats